Computing Persistent Homology of Flag Complexes via Strong Collapses

Jean-Daniel Boissonnat, Siddharth Pritam

Computing Persistent Homology of Flag Complexes via Strong Collapses*

Jean-Daniel Boissonnat
Université Côte d'Azur, INRIA, France
Jean-Daniel.Boissonnat@inria.fr

Siddharth Pritam
Université Côte d'Azur, INRIA, France
siddharth.pritam@inria.fr

Abstract

This paper is a continuation of the research reported in [7] on the usage of strong collapses to accelerate the computation of persistent homology (PH). We show that further decisive progress can be obtained if one restricts the family of simplicial complexes to flag complexes. The resulting method is simple and extremely efficient.

2012 ACM Subject Classification Mathematics of computing, Topological Data Analysis, Computational geometry

Keywords and phrases Computational Topology, Topological Data Analysis, Strong Collapse, Persistent homology

Lines 493

1 Introduction

In this article, we address the problem of computing the Persistent Homology (PH) of a given sequence of simplicial complexes in an efficient way. It is known that computing persistence can be done in $O(n^\omega)$ time, where n is the total number of simplices and $\omega \leq 2.4$ is the matrix multiplication exponent [36, 30]. In practice, when dealing with massive and high-dimensional datasets, n can be very large (of order of billions) and computing PH is then very slow and memory intensive. Improving the performance of PH computation is therefore of utmost importance.

Much progress has been accomplished in the recent years in two directions. First, a number of clever implementations and optimizations have led to a new generation of software for PH computation [31, 45, 5, 38]. Secondly, a complementary direction has been explored to reduce the size of the complexes in the sequence while preserving or approximating in a controlled way the persistent homology of the sequence. Examples are the work of Mischaikow and Nanda [37] who use Morse theory to reduce the size of a filtration, and the work of Dłotko and Wagner who use simple collapses [25]. Both methods compute the exact PH of the input sequence. Approximations can also be computed with theoretical guarantees. Approaches like interleaving with smaller and easily computable simplicial complexes, or sub-sampling the point sample work well upto certain approximation factors [14, 9, 43, 34, 16, 23].

This paper is a continuation of the research reported in [7] on the usage of strong collapses to accelerate the computation of the persistent homology of a sequence of simplicial complexes.

* This research has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007- 2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The basic idea is to simplify the complexes of the input sequence by using strong collapses, as introduced by J. Barmak and E. Miniam [3], and to compute the PH of an induced sequence of reduced simplicial complexes that has the same PH as the initial one. A crucial advantage of the method is that it only needs to store the maximal simplices of the complex, not the full set of the simplices of all dimensions, which saves space and time by a factor that is exponential in the dimension of the complex in the worst-case. As a result and as demonstrated by numerous experiments on publicly available data sets, the approach is very fast and memory efficient in practice.

In this paper, we build on the initial success of [7] and show that further decisive progress can be obtained if one restricts the family of simplicial complexes to flag complexes. Flag complexes are fully characterized by their graph (or 1-skeleton), the other faces being obtained by computing the cliques of the graph. Hence, a flag complex can be represented by its 1-skeleton, which is a very compact representation. Flag complexes are very popular and, in particular, Vietoris-Rips complexes are by far the most widely used simplicial complexes in Topological Data Analysis. It has been shown in [7] that the persistent homology of Vietoris-Rips filtrations can be computed very efficiently using strong collapses. However, most of the time was devoted to computing the maximal cliques of the complex prior to their strong collapse. In this paper, we show that the reduced complex obtained by strong collapsing a flag complex is itself a flag complex. Moreover, this reduced complex can be computed using only the 1-skeleton of the complex and does not require to compute the set of its maximal cliques. Finally, we show how to compute the induced sequence of reduced simplicial complexes using again only the 1-skeleton.

The resulting method is simple and extremely efficient. On the theory side, we show that strong collapses can be computed in time $O(k^2v^2)$ where v is the number of vertices of the complex and k the maximal degree of its graph. The algorithm described in this paper has been implemented. Numerous experiments show that the computation of the persistent homology of flag complexes can be obtained much faster than with previous methods, e.g. Ripser [5]. The code will be soon released in the Gudhi library [31].

2 Preliminaries

In this section, we provide a brief review of the notions of simplicial complex and strong collapse as introduced in [3]. We assume some familiarity with basic concepts like homotopic maps, homotopy type, homology groups and other algebraic topological notions. Readers can refer to [32] for a comprehensive introduction to these topics.

Simplex, simplicial complex and simplicial map : An **abstract simplicial complex** K is a collection of subsets of a non-empty finite set X, such that for every subset A in K, all the subsets of A are in K. From now on we will call an *abstract simplicial complex* simply a *simplicial complex* or just a *complex*. An element of K is called a **simplex**. An element of cardinality $k+1$ is called a k-simplex and k is called its **dimension**. A simplex is called **maximal** if it is not a proper subset of any other simplex in K. A sub-collection L of K is called a **subcomplex**, if it is a simplicial complex itself.

A vertex to vertex map $\psi : K \to L$ between two simplicial complexes is called a **simplicial map**, if the images of the vertices of a simplex always span a simplex. Simplicial maps are thus determined by the images of the vertices. In particular, there is a finite number of simplicial maps between two given finite simplicial complexes. Simplicial maps induce continuous maps between the underlying geometric realisations of the simplicial complexes. Two simplicial

maps $\phi : K \to L$ and $\psi : K \to L$ are **contiguous** if, for all $\sigma \in K$, $\phi(\sigma) \cup \psi(\sigma) \in L$. Two contiguous maps are known to be homotopic [39, Theorem 12.5].

Flag complex: A complex K is a flag or a clique complex if, when a set of its vertices have pair wise edges between them, they span a simplex. It follows that the full structure of K is determined by its 1-skeleton we denote by G. For a vertex v in G, the **open neighborhood** $N_G(v)$ of v in G is defined as $N_G(v) := \{u \in G \mid [uv] \in E\}$. The **closed neighborhood** $N_G[v]$ is $N_G[v] := N_G(v) \cup \{v\}$. We further define the relative closed neighborhood of u by v in G as the set of vertices in $N_G[u]$ that are not in $N_G[v]$. We denote it by $N_G[u \setminus v]$.

Dominated vertex: Let σ be a simplex of a simplicial complex K, the **closed star** of σ in K, $st_K(\sigma)$ is a subcomplex of K which is defined as follows, $st_K(\sigma) := \{\tau \in K |\ \tau \cup \sigma \in K\}$. The **link** of σ in K, $lk_K(\sigma)$ is defined as the set of simplices in $st_K(\sigma)$ which do not intersect with σ, $lk_K(\sigma) := \{\tau \in st_K(\sigma) | \tau \cap \sigma = \emptyset\}$.

Taking a join with a vertex transforms a simplicial complex into a **simplicial cone**. Formally if L is a simplicial complex and a is a vertex not in L then the simplicial cone aL is defined as $aL := \{a, \tau \mid \tau \in L \text{ or } \tau = \sigma \cup a; \text{ where } \sigma \in L\}$. A vertex v in K is called a **dominated vertex** if the link of v in K, $lk_K(v)$ is a simplicial cone, that is, there exists a vertex $v' \neq v$ and a subcomplex L in K, such that $lk_K(v) = v'L$. We say that the vertex v' is dominating v and v is dominated by v'. The symbol $K \setminus v$ (deletion of v from K) refers to the subcomplex of K which has all simplices of K except the ones containing v. Below is an important remark from [3, Remark 2.2], which proposes an alternative definition of dominated vertices.

Remark 1: A vertex $v \in K$ is dominated by another vertex $v' \in K$, *if and only if* all the maximal simplices of K that contain v also contain v' [3].

Strong collapse: An **elementary strong collapse** is the deletion of a dominated vertex v from K, which we denote with $K \searrow\searrow K \setminus v$. Figure 1 illustrates an easy case of an elementary strong collapse. There is a **strong collapse** from a simplicial complex K to its subcomplex L, if there exists a series of elementary strong collapses from K to L, denoted as $K \searrow\searrow L$. The inverse of a strong collapse is called a **strong expansion**. If there exists a combination of strong collapses and/or strong expansions from K to L, then K and L are said to have the same **strong homotopy type**.

The notion of strong homotopy type is stronger than the notion of simple homotopy type in the sense that if K and L have the same strong homotopy type, then they have the same simple homotopy type, and therefore the same homotopy type [3]. There are examples of contractible or simply collapsible simplicial complexes that are not strong collapsible.

A complex without any dominated vertex will be called a **minimal complex**. A **core** of a complex K is a minimal subcomplex $K^c \subseteq K$, such that $K \searrow\searrow K^c$. *Every simplicial complex has a* ***unique core*** *up to isomorphism. The core decides the strong homotopy type of the complex*, and two simplicial complexes have the same strong homotopy type *if and only if* they have isomorphic cores [3, Theorem 2.11].

Retraction map: If a vertex $v \in K$ is dominated by another vertex $v' \in K$, the vertex map $r : K \to K \setminus v$ defined as: $r(w) = w$ if $w \neq v$ and $r(v) = v'$, induces a simplical map that is a *retraction* map. The homotopy between r and the identity $i_{K\setminus v}$ over $K \setminus v$ is in fact a strong deformation retract. Furthermore, the composition $(i_{K\setminus v})r$ is contiguous to the identity i_K over K [3, Proposition 2.9].

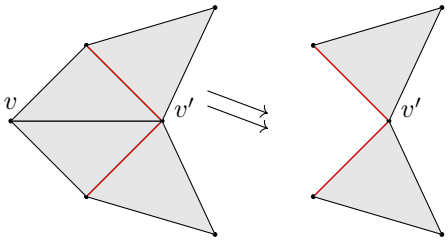


Figure 1 Illustration of an *elementary strong collapse*. In the complex on the left, v is dominated by v'. The link of v is highlighted in red. Removing v leads to the complex on the right.

Persistent homology A **sequence** of simplicial complexes $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} K_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{(m-1)}} K_m\}$, connected through simplicial maps f_is is called a **simplicial tower** or simply a *tower*. We call a tower a **flag tower** if all the simplicial complexes K_i are flag complexes. When all the simplicial maps f_is are inclusions, then the tower is called a filtration and a flag tower will be called a **flag filtration**.

If we compute the homology classes of all the K_i, we get the sequence $\mathcal{P}(\mathcal{T}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xrightarrow{f_2^*} H_p(K_3) \xrightarrow{f_3^*} \cdots \xrightarrow{f_{(m-1)}^*} H_p(K_m)\}$. Here $H_p()$ denotes the homology class of dimension p with coefficients from a field $\mathbb{F}$ and $*$ denotes an induced homomorphism. $\mathcal{P}(\mathcal{T})$ is a sequence of vector spaces connected through homomorphisms, called a **persistence module**. More formally, a *persistence module* $\mathbb{V}$ is a sequence of vector spaces $\{V_1 \to V_2 \to V_3 \to \cdots \to V_m\}$ connected with homomorphisms $\{\to\}$ between them. A persistence module arising from a sequence of simplicial complexes captures the evolution of the topology of the sequence.

Any persistence module can be *decomposed* into a collection of intervals of the form $[i, j)$ [10]. The multiset of all the intervals $[i, j)$ in this decomposition is called the **persistence diagram** of the persistence module. An interval of the form $[i, j)$ in the persistence diagram of $\mathcal{P}(\mathcal{T})$ corresponds to a homological feature (a 'cycle') which appeared at i and disappeared at j. The persistence diagram completely characterizes the persistence module, that is, there is a bijective correspondence between them [10, 49].

Two different persistence modules $\mathbb{V} : \{V_1 \to V_2 \to \cdots \to V_m\}$ and $\mathbb{W} : \{W_1 \to W_2 \to \cdots \to W_m\}$, connected through a set of homomorphisms $\phi_i : V_i \to W_i$ are **equivalent** if the ϕ_i are isomorphisms and the following diagram commutes [10, 20].

$$\begin{array}{ccccccc} V_1 & \longrightarrow & V_2 & \cdots & \longrightarrow V_{m-1} & \longrightarrow & V_m \\ \downarrow{\scriptstyle \phi_1} & & \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \phi_{m-1}} & & \downarrow{\scriptstyle \phi_m} \\ W_1 & \longrightarrow & W_2 & \cdots & \longrightarrow W_{m-1} & \longrightarrow & W_m \end{array}$$

The equivalent persistence modules will have the same interval decomposition, therefore the same diagram.

2.1 Overview of the algorithm

The algorithm presented in this paper adopts the same strategy as the algorithm reported in [7]. By focussing on *flag towers* instead of general sequences of simplicial complexes, we obtain a significantly more efficient algorithm regarding space and time complexity.

Let $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} \cdots \xrightarrow{f_{(m-1)}} K_m\}$ be a flag tower of which we want to compute the persistence diagram. We first compute a *reduced core tower* by computing the cores of each K_i and connecting them through induced simplicial maps as suggested in [7]. Since each K_i is a flag complex, the computation of its core can be computed using only the 1-skeleton of the complex. This is much more efficient than what is done in [7] where the complex is represented by its maximal faces. The new algorithm is discussed in detail in Section 3.

We then compute an associated flag *filtration* we can send to any algorithm that computes the persistence homology of a flag filtration [38, 5, 45, 31]. This is similar to what has been done in [21, 33] and [7], and is detailed in Section 4.

3 Strong Collapse of a Flag complex

In this section, we show that the core of a flag complex K is itself a flag complex whose graph is called the *core graph* of K. The core graph of K can be computed from the 1-skeleton G of K in time $\mathcal{O}(v^2k^2)$, where v is the number of vertices in K and k is an upper bound on the degree of G (i.e. the number of edges that are incident on a vertex in K).

Although this change wrt to the algorithm in [7] might look minor, it is crucial in practice as the time to compute all the maximal simplices of a flag complex from its graph is exponential in the number of its vertices. We thus reduce immensely the time and space complexity of the general algorithm of [7] whose complexity is $\mathcal{O}(v^2\Gamma_0 d + m^2\Gamma_0 d)$, where Γ_0 is an upper bound on the number of *maximal simplices* incident to a vertex.

In the following lemma, we describe a condition in terms of the closed neighborhood $N_G[v]$ of a vertex v of a flag complex K under which v will be dominated by another vertex v' of K. This result has been independently studied in [29, Lemma 4.1].

▶ **Lemma 1.** *Let K be a flag complex. A vertex $v \in K$ is dominated by v' iff $N_G[v] \subseteq N_G[v']$.*

Proof. If v is dominated by v', then, according to Remark 1, all the maximal simplices that contain v also contain v'. Since K is a flag complex, it follows that all edges incident on v are also incident on v'. In other words, $N_G[v] \subseteq N_G[v']$.

Now we prove the other direction. Let σ be a maximal simplex of K containing v. Any other vertex x of σ is joined to v by an edge $[x, v] \in \sigma$. Moreover, since $N_G[v] \subseteq N_G[v']$, $[v, v']$ and $[x, v']$ are in K. It follows that every vertex in σ has an edge with both v and v' and, since K is a flag complex and σ is maximal, v' must be in σ. This implies that all the maximal simplices that contains v also contains v'. Hence v is dominated by v'. ◀

As mentioned before, an elementary strong collapse consists in removing a dominated vertex, and it can be easily observed that removing a vertex does not affect the 'flagness' of the residual complex $K \setminus v$. In other words, if σ is a maximal clique with vertex v, the resultant clique $\sigma \setminus v$ is still a maximal clique in $K \setminus v$. Moreover, all the other cliques that do not contain v still span the complete simplices. This implies that the core K^c of a flag complex K with graph G is a flag complex of a graph G^c of G.

In what follows next, we describe an algorithm to compute the core graph $G^c \subseteq G$ whose flag complex is the core K^c of K.

Data structure: We represent G with its adjacency matrix M, where the rows and the columns of M represent the vertices of G. An entry $M[v_i][v_j]$ associated with vertices v_i and v_j is set to 1 if either the edge $[v_i, v_j] \in G$ or $i = j$, and to 0 otherwise. We will say that a row v is contained in another row v' if the set of indices of the non-zero entries of v is a

subset of the indices of the non-zero entries of $v\prime$. It is clear that if a row v is contained in another row v', we have $N_G[v] \subseteq N_G[v']$ and therefore the vertex v is dominated by the vertex v'

Core algorithm: Given the adjacency matrix M of G, we compute the adjacency matrix C of the core graph G^c. In view of Lemma 1, we can easily compute C from M using basic row removal operations. Loosely speaking, we remove the rows of M that are contained in another row. After removing the row associated to v, we simultaneously update the matrix by removing the column associated to v. The process is iterated as long as the matrix can be reduced. Upon termination, we output the reduced matrix C, which is the adjacency matrix of the core graph G^c of K. Since the core of a complex is always unique, the order in which vertices are removed does not matter [3].

Computing the retraction map r: The algorithm also provides a direct way to compute the retraction map r defined in Section 2. It can be constructed as follows. A row v being removed in M corresponds to a dominated vertex in K and the row which contains v corresponds to a dominating vertex. Therefore we map the dominated vertex to the dominating vertex.

Reducing the number of domination tests: We first observe that, when one wants to determine if a row v is dominated by some other row v', we don't need to test v with all other rows but only with its neighbors, i.e. at most k of them. Here k is the upper bound on the degree of the vertices in G.

A second observation is that we don't need to test all rows for domination, but only the so-called candidate rows. We define a row v to be a **candidate row** for the next iteration if at least one of its neighbors has been removed in a previous row removal iteration. Candidate rows are the only rows that need to be considered in the domination tests of the algorithm. Indeed, a row w of M whose neighbors are present from the previous *iteration* cannot be dominated by another row v' of M, as w was not dominated in the previous iteration and all other vertices can only loose their neighbors. This ensures that v will still remain un-dominated.

We maintain a *queue*, for the candidate rows (rowQueue). These queues are implemented as First in First out (FIFO) queues. At each iteration, we *pop out* a candidate row from rowQueue and test whether it is dominated or not. After each successful domination test, we push the new candidate rows in the queue in preparation for the subsequent iteration. In the first iteration, we push all the rows in rowQueue. Algorithm 1 gives the pseudo code of our algorithm.

Time Complexity: The most basic operation in our algorithm is to determine if a row is dominated by another given row. In our implementation, the rows of the matrix that are considered by the algorithm are stored as sorted lists. Checking if one sorted list is a subset of another sorted list can be done in time $\mathcal{O}(l)$, where l is the size of the longer list. Note that the length of a row list is at most $k+1$ where k denotes an upper bound on the degree of any vertex. Hence checking if a row is dominated by another row takes $\mathcal{O}(k)$ time.

At each iteration on the rows (Lines 6-12 of Algorithm 1), each row is checked against at most k other rows (maximum number of neighbors). Moreover, since at each iteration on the rows we remove at least one row, the total number of iterations on the rows is at most

Algorithm 1 Core algorithm

```
procedure CORE(M) ▷ Returns the adjacency matrix corresponding to the core of K
    rowQueue ← push all rows of M (all vertices of K)
    while rowQueue is not empty do
        v ← pop(rowQueue)
        N_G[v] ← the non-zero columns of v
        for w in N_G[v] do
            if N_G[v] ⊆ N_G[w] then
                Remove v from M ▷ Both the associated column and the associated row
                push all the entries of N_G(v) to rowQueue if not pushed before
                break
            end if
        end for
    end while
    return M ▷ The core consists of the remaining rows and columns
end procedure
```

$O(v^2)$, where v is the total number of vertices of the complex K. Therefore, the worst-case time complexity of our algorithm is $\mathcal{O}(v^2k^2)$.

4 Flag Tower to Flag Filtration

In this section, we show that, thanks to the notion of strong collapses, we can efficiently turn a flag *tower* into a flag *filtration* using only edge inclusions over the 1-skeletons of the complexes.

4.1 Previous work

It is well known that any general simplicial map can be decomposed into more elementary simplicial maps, namely elementary inclusions (i.e., inclusions of a single simplex) and elementary contractions (where a vertex is mapped onto another vertex). It can be observed that if we can replace an elementary contraction $\{\{u, v\} \mapsto u\}$ with an equivalent (not necessarily elementary) inclusion, we thus transform a tower into an equivalent filtration. This was the philosophy introduced by Dey et. al. in [21]. Given an elementary contraction $K_i \xrightarrow{\{u,v\}\mapsto u} K_{i+1}$, their strategy was to fill the simplices around the edge $[uv]$ such that it satisfies the *link condition*. Let $\hat{K}_{i+1} := \{K_i \cup S\}$, where S contains the missing simplices around $[uv]$ such that $[uv]$ satisfies the link condition in $\hat{K}_{i+1}$. The crucial observation is that including S does not create new topological changes. More precisely, the following diagram commutes and the inclusion $(i')^* : H_p(K_{i+1}) \hookrightarrow H_p(\hat{K}_{i+1})$ is an isomorphism.

$$\begin{array}{ccc} H_p(K_i) & \xrightarrow{f_i^*} & H_p(K_{i+1}) \\ & \searrow^{i^*} & \downarrow{\scriptstyle (i')^*} \\ & & H_p(\hat{K}_{i+1}) \end{array}$$

Applying this construction as many times as required, we can see that any tower can be transformed into an equivalent filtration.

The work of Kerber and Schreiber [33] uses a slightly different approach where instead of the link condition they use a coning strategy. They define $\hat{K}_{i+1} := \{K_i \cup (u * St_{K_i}(v))\}$. The differently defined $\hat{K}_{i+1}$ and the associated inclusion map $(i')^*$ also satisfy the aforementioned commutativity and isomorphism. However, if one uses the coning strategy naively, the size of the final resultant filtration may not be optimal. They address this issue using two crucial observations. They first observe that one does not need to cone u with the complete star $st_{K_i}(v)$ of v but only with a subset $ActSt_{K_i}(v)$ of it, called an active star. The **active star** $ActSt_{K_i}(v)$ of a vertex v in K_i is the set of simplices in the star $st_{K_i}(v)$ that doesn't contain any vertex whose star has been coned before. This is implemented as follows. Initially, all vertices are marked as active. Then, after the coning $\hat{K}_{i+1} := \{K_i \cup (u * St_{K_i}(v))\}$, the vertex v is marked *inactive*. They also observed that mapping $\{\{u, v\} \mapsto u\}$ or $\{\{u, v\} \mapsto v\}$ yields isomorphic simplicial complexes $\hat{K}_i$ upto renaming the vertices. Choosing the representative whose active star is smaller can lead to a smaller complex $\hat{K}_{i+1}$. Finally, they prove that, given a tower $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$ of elementary inclusions and elementary contractions, the size of the equivalent filtration is $\mathcal{O}(d * n * \log n_0)$, where d is the maximal dimension of the K_is, n the total number of elementary inclusions in $\mathcal{T}$ and n_o is the number of vertices included in $\mathcal{T}$ [33, Theorem 2]. Further they show that the time and space complexities of their algorithm are $O(d * |\hat{K}_m| * \mathbb{C}_\omega)$ and $\mathcal{O}(d * \omega)$ respectively, where $\mathbb{C}_\omega$ is the cost of an operation in a dictionary with ω elements, where ω is the maximal number of simplices in K_is [33, Theorem 2].

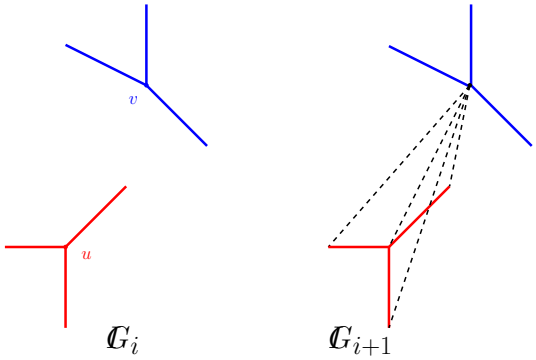
4.2 A new construction

Our work builds upon the above mentioned previous works [21, 33]. The difference is that we use strong expansions instead of coning, a strong expansion being the inverse operation of a strong collapse. The main advantage of strong expansions is that, when the input is a flag tower, we can use the domination criterion of Lemma 1. This leads to a simple algorithm that only deals with edges. The final filtration is a flag filtration, which can be represented very compactly. Moreover, a strong expansion being a coning, we will be able to use the theoretical results of [33]. Now we describe our construction.

Let K_i be a flag complex and G_i be its 1-skeleton for $i = 0, ..., m$. As in [21, 33], we associate to K_i an augmented complex noted $\mathbb{K}_i$ which plays a role similar to $\hat{K}_i$ in the previous subsection. As will be seen below, $\mathbb{K}_i$ is also a flag complex whose 1-skeleton will be denoted by $\mathbb{G}_i$. Following the terminology of [33], we call a vertex $v \in \mathbb{K}_i$ to be **active** if it is currently not *dominated* and has never been *contracted* before. The active closed neighborhood $ActN_{G_i}[v]$ is then defined as the set of all active vertices in $N_{G_i}[v]$. Similarly, $ActN_{G_i}[v \setminus u]$ denotes the set of active vertices in the closed neighborhood $N_{G_i}[v]$ of v that are not in $N_{G_i}[u]$. Finally, let $\{[u, ActN_{G_i}[v \setminus u]]\}$ denote the set of edges between u and $ActN_{G_i}[v \setminus u]$.

Using the notions defined above, we now explain how to inductively construct a filtration associated to a given flag tower. For $i = 0$, we set $\mathbb{G}_0 = \emptyset$. We then define $\mathbb{G}_i$ as follows.

- if $G_i \xrightarrow{\cup\sigma} G_{i+1}$ is an elementary *inclusion* where σ is either a vertex or an edge, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \sigma$.
- if $G_i \xrightarrow{\{u,v\}\mapsto u} G_{i+1}$ is an elementary *contraction*
  - if $|ActN_{\mathbb{G}_i}[v \setminus u]| \leq |ActN_{\mathbb{G}_i}[u \setminus v]|$, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[u, ActN_{\mathbb{G}_i}[v \setminus u]\}$ and v as contracted
  - otherwise, we set $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[v, ActN_{\mathbb{G}_i}[u \setminus v]\}$ and u as contracted.



There are two different kinds of inactive vertices; the ones that have been marked as *contracted* and the others which are currently dominated. The first type is permanent while the second type may change over time. Observe that the construction checks for domination only before the elementary contractions not before the elementary inclusions. Therefore, before each such contraction $\{u, v\} \mapsto u$, we visit all the vertices to see if it is dominated in $\mathbb{G}_i$. In fact, we only consider the vertices that have gained new edges after the previous contraction. Indeed, they are the only ones whose neighborhood has changed and can start dominating other vertices. To implement this optimization, we maintain and update a flag for each vertex that indicates whether a new edge became incident on it after the last contraction. We continue the construction until the end of our input tower.

▶ **Lemma 2.** *Let* $f_i : K_i \xrightarrow{\{u,v\}\mapsto u} K_{i+1}$ *be the first elementary contraction in the tower* $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{m-1}} K_m$. *Then the complex* K_{i+1} *is a subcomplex of* $\mathbb{K}_{i+1}$ *and* $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$.

Proof. We prove the second part $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$ of the statement which then implies the first part $K_{i+1} \subset \mathbb{K}_{i+1}$. Since f_i is the first contraction $\mathbb{K}_i = K_i$ and $\mathbb{G}_i = G_i$. Let $\mathbb{G}_{i+1} := G_i \cup \{[u, ActN_{G_i}[v \setminus u]\}$ be the graph defined above. By construction, contracting the pair $\{u, v\}$ to u in both the graphs G_i, $\mathbb{G}_{i+1}$ yields the same graph G_{i+1}.

Let $x' \in ActN_{G_i}[v \setminus u]$. We observe that adding the edge $[ux']$ in G_i does not change the domination of $x \in \{N_{G_i}[v \setminus u] \setminus ActN_{G_i}[v \setminus u]\}$ as it only add neighbors in $N_{G_i}[x']$ and $N_{G_i}[u]$. Therefore, adding the edges $\{[u, ActN_{G_i}[v \setminus u]\}$ to G_i does not change the domination status of the vertices of $\mathbb{K}_{i+1}$ that are in the set $\{N_{G_i}[v \setminus u] \setminus ActN_{G_i}[v \setminus u]\}$. Therefore, removing all the dominated vertices in $N_{G_i}[v \setminus u]$ provides a sequence of elementary strong collapses. By performing all such elementary strong collapses, $\mathbb{K}_{i+1}$ is eventually transformed into a complex K^0_{i+1}. Moreover, we have removed all the dominated vertices from $N_{G_i}[v \setminus u]$ and added edges between u and the remaining vertices in $ActN_{\mathbb{G}_i}[v]$. This implies that v is dominated by u in K^0_{i+1}. Collapsing v onto u, implies $K^0_{i+1} \searrow\searrow K_{i+1}$ and therefore $\mathbb{K}_{i+1} \searrow\searrow K_{i+1}$.

◀

▶ **Lemma 3.** *Let* $f_i : K_i \xrightarrow{\{u,v\}\mapsto u} K_{i+1}$ *be the first elementary contraction in the tower* $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{m-1}} K_m$. *Then the following diagram commutes*

$$\begin{array}{ccc} H_p(K_i) & \xrightarrow{f_i^*} & H_p(K_{i+1}) \\ & \searrow^{i^*} & \downarrow{\scriptstyle (i')^*} \\ & & H_p(\mathbb{K}_{i+1}) \end{array}$$

Here $i' : K_{i+1} \hookrightarrow \mathbb{K}_{i+1}$ is the inclusion induced by the strong collapse. i^* and $(i')^*$ are homomorphisms induced by the inclusion maps.

Proof. As mentioned before, since f_i is the first contraction $\mathbb{K}_i = K_i \subseteq \mathbb{K}_{i+1}$. Let K^0_{i+1} be the complex as defined in the proof of Lemma 2. Consider the following diagram of the simplicial complexes, and note that $i' = i_1 \circ i_0$ where both i_0 and i_1 are inclusions induced by the respective strong collapses.

$$\begin{array}{ccc} K_i & \xrightarrow{f_i} & K_{i+1} \\ \downarrow{\scriptstyle i} & & \downarrow{\scriptstyle i_0} \\ \mathbb{K}_{i+1} & \underset{i_1}{\hookleftarrow} & K^0_{i+1} \end{array}$$

We claim that the maps $i' \circ f_i$ and i are contiguous, which we denote $i' \circ f_i \sim i$. Indeed, let σ be any simplex in K_i. Since i is an inclusion, $i(\sigma) = \sigma$. If $v \notin \sigma$, then $i' \circ f_i(\sigma) = \sigma = i(\sigma)$. Hence if $v \notin \sigma$, $i' \circ f_i \sim i$.

If $v \in \sigma$, $f_i(\sigma)$ is a simplex $\gamma \in K_{i+1}$ that contains u and, since i_0 is an inclusion, $i_0 \circ f_i(\sigma) = \gamma$. Observe that, in the retraction map associated to the strong collapse $r_1 : \mathbb{K}_{i+1} \searrow\searrow K^0_{i+1}$, v is not contracted (by construction of K^0_{i+1}). Therefore $r_1 \circ i(\sigma)$ is a simplex $\gamma' \in K^0_{i+1}$ containing v.

Now, as mentioned in the proof of Lemma 2, u dominates v in K^0_{i+1}. Therefore all the maximal simplices in K^0_{i+1} that contain v also contain u. Therefore, γ' will be a face of a maximal simplex $\tau \in K^0_{i+1}$ that contains u.

Since γ is obtained by contracting v to u, γ must be a face of such a $\tau \in K^0_{i+1}$, which contains both u and v. This implies that $\gamma' \cup \gamma \subseteq \tau$ for some maximal simplex $\tau \in K^0_{i+1}$, which in turn implies that $r_1 \circ i(\sigma)$ is contiguous to $i_0 \circ f_i(\sigma)$. After composing both sides with i_1 we get $i_1 \circ r_1 \circ i(\sigma) \sim i_1 \circ i_0 \circ f_i(\sigma)$. Now since $\mathbb{K}_{i+1} \searrow\searrow K^0_{i+1}$, $i_1 \circ r_1 \sim 1_{\mathbb{K}_{i+1}}$ [3], where $1_{\mathbb{K}_{i+1}}$ is the identity over $\mathbb{K}_{i+1}$. As $i_1 \circ i_0 = i'$, we conclude $i' \circ f_i \sim i$.

Since contiguous maps are homotopic at the level of geometric realizations, the diagram in the lemma commutes. ◀

The following lemma is a more general version of Lemma 2 stated above. Its proof follows from simple inductive arguments.

▶ **Lemma 4.** *Given a tower $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{m-1}} K_m$. for each $0 \leq i \leq m$, $\mathbb{K}_i \searrow\searrow K_i$.*

Again using an inductive argument along with Lemmas 3 and 4, we can deduce the following result.

▶ **Theorem 5.** *The following diagram commutes and therefore the tower $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{m-1}} K_m$ and the constructed filtration $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow \ldots \hookrightarrow \mathbb{K}_m$ have the same persistence diagram.*

$$\begin{array}{ccccccccc} H_p(\mathbb{K}_1) & \xrightarrow{*} & H_p(\mathbb{K}_2) & \xrightarrow{*} & \cdots & \longrightarrow & H_p(\mathbb{K}_{m-1}) & \xrightarrow{*} & H_p(\mathbb{K}_m) \\ \downarrow{\scriptstyle \phi_1^*} & & \downarrow{\scriptstyle \phi_2^*} & & & & \downarrow{\scriptstyle \phi_{m-1}^*} & & \downarrow{\scriptstyle \phi_m^*} \\ H_p(K_1) & \xrightarrow{f_1^*} & H_p(K_2) & \xrightarrow{f_2^*} & \cdots & \longrightarrow & H_p(K_{m-1}) & \xrightarrow{f_{m-1}^*} & H_p(K_m) \end{array}$$

Here ϕ_i is a strong collapse for each $i \in \{0, \cdots, m\}$ and $*$ indicates the induced homomorphisms.

Complexity Analysis: The *contracted* vertices defined here are exactly the same as the inactive vertices in [33]. By construction, any contracted vertex will be dominated permanently in the filtration. As such a vertex stops being existent in the tower later on, its neighborhood stays the same and the vertex remains dominated. Therefore, all the *active* vertices in our construction are the vertices that are currently not dominated. However, we choose to differentiate the two different types of inactive vertices (dominated and contracted) to emphasise that, at any point in our construction, the number of active vertices is less than the number of active vertices that are used in [33]. Moreover, since a strong expansion is a coning, the size of the final filtration in our construction is at most that obtained by the construction prescribed in [33]. Moreover, since we are working with 1-skeletons only, the space and time complexity of our method is much lower than that of [33].

To analyze the time complexity, observe that each edge inclusion can be performed in constant time $\mathcal{O}(1)$ and, before each contraction, the domination relationships can be updated in $\mathcal{O}(v * k^2)$ time. Note that we are not computing the core here we are just computing the currently dominated vertices. For each strong expansion in the augmented graphs corresponding to the contraction $(\{w, u\} \mapsto u)$ in the original graph, we look at the set-difference between at most k neighbors of the two vertices. This can be performed in $\mathcal{O}(k)$ time. Therefore, given a tower with n_c elementary contractions, and $|\mathbb{G}_m|$ the size of the skeleton of the final equivalent flag filtration can be computed in at most $\mathcal{O}(|\mathbb{G}_m| + n_c * v * k^3)$. The space complexity of our construction is $\mathcal{O}(n_0 * k)$ which is the size of the sparse adjacency matrix of the final flag filtration of n_0 vertices. We summarize our result in the following theorem.

▶ **Theorem 6.** *Let $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} ... \xrightarrow{f_{m-1}} K_m$ be a flag tower where, w.l.o.g., $K_0 = \emptyset$ and each f_i is either an inclusion (not necessarily elementary but corresponds to an elementary inclusion on the graphs G_i) or an elementary contraction. Let d denote the maximal dimension of K_is in $\mathcal{T}$, and let n denote the total number of elementary inclusions of simplices in $\mathcal{T}$, n_c total number of elementary contraction and n_0 the number of vertex inclusions in $\mathcal{T}$. Then, there exists a filtration $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow ... \hookrightarrow \mathbb{K}_m$, where the inclusions are not necessarily elementary, such that $\mathcal{T}$ and $\mathcal{F}$ have the same persistence diagram and the size of the filtration $|\mathbb{K}_m|$ is at most $\mathcal{O}(d * n * \log n_0)$. Moreover, $\mathcal{F}$ is a filtered flag complex and can be computed from $\mathcal{T}$ using only the 1-skeletons G_is of K_is in $O(|\mathbb{G}_m| + n_c * n_0 * k^3)$ time and $\mathcal{O}(n_0 * k)$ space complexity, here k is the upper bound on the degree of the vertices in $\mathbb{G}_m$.*

5 Computational experiments

We compute the persistence diagram (PD) of VR-filtrations associated to different data. The filtration value of a simplex in a VR filtration is the length of the longest edge of the simplex.

Approximate persistence diagram Given a VR filtration, one can choose to collapse the original complexes after each edge inclusion. However, we can also choose to strong collapse the complexes less often, i.e. after several edge inclusions rather than just one. This will result in a faster algorithm but comes with a cost: the computed PD is then only approximate. We call **snapshots** the values of the scale parameter at which we choose to strong collapse the complex. The difference between two consecutive snapshots is called a **step**. We approximate the filtration value of a simplex as the value of the snapshot at which it first appears. We can observe that our algorithm will report all persistence pairs that are separated by at least one snapshot. Hence if all steps are equal to some $\epsilon > 0$, we will compute all the persistence pairs

whose lengths are at least ϵ. It follows that the *bottleneck distance* between the computed PD and the exact one is at most ϵ.

Experimental setup Now we present some experimental results comparing our software named RipsCollapser with Ripser [5], which is the state of the art software to compute the PD of VR filtrations. RipsCollapser has been coded in C++. The code has been compiled using the compiler 'clang-900.0.38' and all computations were performed on a '2.8 GHz Intel Core i5' machine with 16 GB of available RAM.

The comparison is done on three datasets **netw-sc**, **senate** and **eleg** from [18]. RipsCollapser takes as input a VR filtration, constructs the associated collapsed sequence and then computes the equivalent flag filtration. The output filtration is then sent to the Gudhi library to compute persistence. The reported time is the total time which includes: 1. The time taken to compute the largest 1-skeleton associated to the maximum threshold value, 2. The time taken to collapse all the sub-skeletons and assemble their cores. 3. To transform them into an equivalent flag-filtration. 4. To compute the PD of the equivalent flag-filtration. We also note the time taken from 1 to 3 as pre-process time.

As mentioned above, we approximate the filtration value of a simplex as the value of the snapshot parameter at which it appears for the first time, whereas, in the case of Ripser, it is the length of the longest edge (1-simplex) it contains. Therefore, the computed PD by RipsCollapser is not exactly the same as the one computed by Ripser. However, in the above experiments, we choose steps that are very small so that the bottleneck distance between the two PD returned by Ripser and RipsCollapser for a given data set is also very small.

Command <./ripser inputData –format distances –threshold inputTh –dim inputDim > was used to run Ripser and we used the distance matrix format for all the datasets. Both RipsCollapser and Ripser have a parameter –dim until which they compute the PD. For a given *threshold* (maximum scale parameter) and *dim*, Ripser computes the *dim*-skeleton of the VR complex and then computes the PD of the *dim*-skeleton. Differently, RipsCollapser computes the 1-skeleton of the VR complex for the maximum scale parameter (Threshold). During the preprocessing, the dimension of the original complex doesn't come into consideration. Therefore the preprocessing is done for all the dimensions. However, we can restrict the dimension (using parameter –*dim*) of the smaller equivalent flag filtration. As the experiments will show, this does not matter much since the collapsed cores have small size and dimension.

Results Table 1 contains the results of the experiments using RipsCollapser and Table 2 contains the results using Ripser. Ripser performs quite well for computing PD for low values of *dim*. However, as we move to intermediate values, it slows down quite considerably and in some cases (*dim* above 7), the size of the complex is so huge that Ripser crashed due to memory overload. Differently RipsCollapser is not much affected by the choice of *dim* both in terms of space and time and we can compute PD for large values of the threshold and of *dim*.

As an additional remark, we note that, in our current implementation, the collapses are performed in sequence. A further improvement would be to perform them in parallel.

RipsCollapser will be available as an open-source package of a next release of the Gudhi library [31].

Data	Pnt	Threshold	RipsCollapser(Gudhi)				
			Dim	Preprocess-Time	Total-Time	Steps	TotSnaps
netw-sc	379	4.5	5	4.99s	5.09s	0.02	213
netw-sc	379	4.5	all	4.99s	5.08s	0.02	213
”	”	5.5	6	9.55s	9.65s	0.02	263
”	”	5.5	all	9.46s	9.56s	0.02	263
senate	103	0.415	all	2.71s	2.73s	0.001	403
eleg	297	0.3	5	11.14s	13.32s	0.001	284
eleg	297	0.3	all	11.15s	25.3s	0.001	284

Table 1 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum scale parameter (Threshold), dimension of the collapsed flag-Complex (Dim), time taken to collapse and compute equivalent flag-filtration (Preprocess-Time), total time taken by RipsCollapser (Gudhi) (Total-Time), incremental steps of subcomplexes (Steps) and total number of snapshots used (TotSnaps). All times are averaged over five trials

Data	Pnt	Threshold	Val		Val		Val	
			Dim	Time	Dim	Time	Dim	Time
netw-sc	379	4.5	4	3.8s	5	21.5s	7	357s
”	”	5.5	4	25.3s	5	231.2s	6	∞
senate	103	0.415	3	0.52s	4	5.9s	5	52.3s
”	”	”	6	406.8s	7	∞		
eleg	297	0.3	3	8.9s	4	217s	5	∞

Table 2 The columns are, from left to right: dataset (Data), number of points (Pnt), maximum scale parameter (Threshold), input dimension for Ripser (Dim), total time taken by Ripser (Time). Most results are averaged over five trials except the longer ones. ∞ in the Time column means that the experiment ran longer than 12hrs or crashed due to memory overload.

References

1 M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry: Theory and Applications*, 57:8–18, 2016.

2 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *International Journal of Computational Geometry and Applications (IJCGA)*, 22:279–303, 2012.

3 J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete and Computational Geometry*, 47:301–328, 2012.

4 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization*, pages 103–117. 2014.

5 Ulrich Bauer. Ripser. URL: `https://github.com/Ripser/ripser`.

6 Jean-Daniel Boissonnat, C. S. Karthik, and Sébastien Tavenas. Building efficient and compact data structures for simplicial complexes. *Algorithmica*, 79:530–567, 2017.

7 Jean-Daniel Boissonnat, Siddharth Pritam, and Divyansh Pareek. Strong Collapse for Persistence. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, 2018.

8 Jean-Daniel Boissonnat and Karthik C. S. An efficient representation for filtrations of simplicial complexes. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.

9 M. Botnan and G Spreemann. Approximating persistent homology in euclidean space through collapses. *In: Applicable Algebra in Engineering, Communication and Computing*, 26:73–101.

10 Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Found Comput Math*, 10, 2010.

11 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. *SOCG*, pages 247–256, 2009.

12 Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *In: International Journal of Computer Vision*, 76:1–12, 2008.

13 Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. Topology of viral evolution. *In: Proceedings of the National Academy of Sciences*, 110:18566–18571, 2013.

14 F. Chazal and S. Oudot. Towards persistence-based reconstruction in euclidean spaces. *SOCG*, 2008.

15 C. Chen and M. Kerber. Persistent homology computation with a twist. *In European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.

16 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra:. In *Polynomial-Sized Topological Approximations Using The Permutahedron*. 32nd International Symposium on Computational Geometry (SoCG), 2016.

17 David Cohen-Steiner and Herbert and Edelsbrunner. Harer,john. stability of persistence diagrams. *Discrete Comput. Geom*, 37:103–120, 2007.

18 Datasets. URL: `https://github.com/n-otter/PH-roadmap/''`.

19 Vin de Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *In: Algebraic and Geometric Topology*, 7:339 – 358, 2007.

20 Harm Derksen and Jerzy Weyman. Quiver representations. *Notices of the American Mathematical Society*, 52(2):200–206, February 2005.

21 T. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In *Symposium on Computational Geometry (SoCG*, pages 345–354, 2014.

22 T. K. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l'Institut Mathematique (Beograd)*, 60:23–45, 1999.

23 Tamal Dey, Dayu Shi, and Yusu Wang. *SimBa: An efficient tool for approximating Rips-filtration persistence via Simplicial Batch-collapse*. In European Symp. on Algorithms (ESA), pages 35:1–35:16, 2016.

24 C. H. Dowker. Homology groups of relations. *The Annals of Mathematics*, 56:84–95, 1952.

25 P. Dłotko and H. Wagner. Simplification of complexes for persistent homology computations,. *Homology, Homotopy and Applications*, 16:49–63, 2014.

26 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom*, 28:511–533, 2002.

27 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

28 Brittany Terese Fasy, Jisu Kim, Fabrizio Lecci, and Clément Maria:. Introduction to the r package tda. *CoRR abs/1411.1830*, 2014.

29 E. Fieux and J. Lacaze. Foldings in graphs and relations with simplicial complexes and posets. *Discrete Mathematics*, 312(17):2639 – 2651, 2012.

30 François Le Gall. Powers of tensors and fast matrix multiplication. *ISSAC '*, 14:296–303, 2014.

31 Gudhi: Geometry understanding in higher dimensions. URL: `http://gudhi.gforge.inria.fr/`.

32 A. Hatcher. *Algebraic Topology*. Univ. Press Cambridge, 2001.

33 Michael Kerber and Hannah Schreiber:. Barcodes of towers and a streaming algorithm for persistent homology. *33rd International Symposium on Computational Geometry*, 2017. arXiv:1701.02208.

34 Michael Kerber and R. Sharathkumar. Approximate Čech complex in low and high dimensions. In *Algorithms and Computation*, pages 666–676. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science, 2013.

35 C. Maria and S. Oudot. Zigzag persistence via reflections and transpositions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA) pp.* 181–199, January 2015.

36 Nikola Milosavljevic, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In *Symposium on Computational Geometry (SoCG)*, 2011.

37 K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *DCG*, 50:330–353, September 2013.

38 Dimitriy Mozozov. Dionysus. URL: http://www.mrzv.org/software/dionysus/.

39 J. Munkres. *Elements of Algebraic Topology*. Perseus Publishing, 1984.

40 N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science, Springer Nature*, page 6:17, 2017.

41 Jose Perea and Gunnar Carlsson. A klein-bottle-based dictionary for texture representation. *In: International Journal of Computer Vision*, 107:75–97, 2014.

42 Hannah Schreiber. Sophia. URL: https://bitbucket.org/schreiberh/sophia/.

43 Donald Sheehy. Linear-size approximations to the vietoris–rips filtration. *Discrete and Computational Geometry*, 49:778–796, 2013.

44 M. Tancer. Recognition of collapsible complexes is np-complete. *Discrete and Computational Geometry*, 55:21–38, 2016.

45 J.Reininghausc U. Bauer, M. Kerber and Hagner:. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78, 2017.

46 J. H. C Whitehead. Simplicial spaces nuclei and m-groups. *Proc. London Math. Soc*, 45:243–327, 1939.

47 A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. *ICASSP - Proceedings*, pages 11–15, 2014.

48 A. C. Wilkerson, T. J. Moore, and and A. H. Krim A. Swami. *Simplifying the homology of networks via strong collapses.* ICASSP - Proceedings, 2013.

49 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom*, 33:249–274, 2005.

50 Afra Zomorodian. The tidy set: A minimal simplicial set for computing homology of clique complexes. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, pages 257–266, SoCG'10. Snowbird, Utah, USA . isbn, 2010.